AutoDrop: Automatic DDoS Detection and Its Mitigation with Combination of OpenFlow and sFlow

Faisal Shahzad[1,2], Muazzam A. Khan[1(✉)], Shoab A. Khan[1], Saad Rehman[1], and Monis Akhlaq[2]

[1] College of EME, National University of Science and Technology, Islamabad, Pakistan
ifaisalshahzad@gmail.com, {muazzamak,shoabak}@ce.ceme.edu.pk
[2] Deltasoft Technologies, Rawalpindi, Pakistan

Abstract. World is emerging into global village with the support of internet connectivity. With the help of this connectivity, it also made everyone subject of being compromised. Many organizations' confidential data and numerous online services become victim of cyber-attacks. Different researches and innovations have been made for making network secure but commercial routers limit them to deploy custom security algorithms in real network. Recently, researchers succeed to innovate a novel protocol OpenFlow in Software Defined Networks. Taking advantage of this innovation we utilized OpenFlow to analyze real-time traffic, detect DDoS attack and mitigate attack. In this paper, we proposed a methodology to automatically detect different type of DDoS attacks within few seconds of occurrence using sampling techniques for continuous monitoring site-wide traffic and block attacking source with the help of OpenFlow protocol.

Keywords: DDoS · SDN · OpenFlow · sFlow · Security

1 Introduction

Many network applications, now-a-days are real time in nature. In a real time application, a good response time (efficient) and high latency rate of users requests are expected. Meeting the users expectations and needs of efficient response time is a major issue to be addressed in real time networks, where the traffic rate is also high, but it becomes really difficult to maintain a reasonable traffic flow in such applications when the unwanted software attacks are thrown to the system. These unwanted attacks make a network system slow or sometimes totally unavailable for its intended users. Our research objective is to address this Distributed Denial of Service (DDoS) problem in Software Defined Network (SDN) architecture.

Software Defined Network (SDN) has emerged in last two decades since 1990. Researchers were eager to make network programmable. In start, from 1990 to

J. Ferreira and M. Alam (Eds.): Future 5V 2016, LNICST 185, pp. 112–122, 2017.
DOI: 10.1007/978-3-319-51207-5_11

2000, many small functions/scripts were developed to automate the network. From 2000 to 2007, researchers focused on separating control plane and data plane. In 2006, Martin Casado, PhD student at Stanford University in Silicon Valley developed something called Ethan. Later on, Stanford and University of California, Berkeley did a joint research and standardize protocol with the name of OpenFlow [1].

OpenFlow is an open standard that enables researchers to run experimental protocols in the campus networks. OpenFlow is used at commercial level and implemented by different vendors. It is embedded in different Ethernet switches, routers and wireless access points. It provides a research platform to run experiments without exposing the networks internal details. OpenFlow facilitates the researchers to innovate routing and switching protocols in networks. OpenFlow commercially being utilized in many applications like virtual machine mobility, high security networks and next generation IP based mobile networks [2].

2 Literature Review

Yao et al., proposed Virtual Source Address Validation Edge (VAVE), a method for securing network from spoofed IPs. They propose the use of OpenFlow devices rather than SAVI devices [3]. They pointed out that SAVI devices are good to detect spoofed IPs but each SAVI device is working independently without collaborating or using other device knowledge because they cannot communicate with each other, eventually which cause recalculation on each device again and again. In their solution, they were used OpenFlow devices with central controller and form a perimeter, when packets enter to perimeter, first OpenFlow device apply validation filters on packet using NOX controller, remaining all just used that information. Shin et al., proposed CloudWatcher, another network security solution that differ from VAVE. They proposed a simple scripting language to help network operators in defining policies. In this design, OpenFlow controller does not have any security module; instead they used other appliance for network security such as intrusion detection system. In this system, OpenFlow captures incoming network packets and forwards them to security appliances that inspect all of them [3]. Instead of OpenFlow controller, Kumar et al., proposed an OpenFlow switch for intrusion detection by adding two more tables for attacker IP (IDS IP) and signatures of malicious attacks. First they look into IDS IP table, if they found packet IP there then they simply drop the packet. If match is not found they look into IDS signatures table for checking its packet malicious or not. If packet found malicious they add it to IDS IP table for future use [4]. This solution greatly helps in: (1) Real-time packet inspection and validation, (2) Real-time attack mitigation, (3) Secure and intelligent network. On the other hand, this solution also impacts on: (1) Number of packets processing per second, (2) Traffic flow per second, (3) Cause bottle-neck in the network, (4) Significant impact on network performance.

Li et al. proposed DrawBridge, based on the assumption that if customer's controller can communicate with ISP controller. This enables customers to

subscribe for ISP's traffic engineering service. Customer express its traffic engineering policies to DrawBridge controller in ISP, the DrawBridge controller further pushes these policies to SDN switches deployed in ISP network to filter traffic or another DrawBridge controller in the ISP upstream [6]. They preferred to throttle traffic rather than block attacking node and also suggest to install rules on ISP hosted OpenFlow controller. They are throttling traffic using OpenFlow protocol instead of dropping useless traffic. By this way useless traffic will be still coming in network and will be making resources busy in useless work. Based on same assumptions as DrawBridge, Sahay et al. proposed autonomic DDoS mitigation using software defined network. They proposed that DDoS mitigation can be autonomous application, as a service which can be consumed by customer network and multiple ISP networks. Both customer and ISP should have their own DDoS detection engine, can alert each other about attacks and use middle box mitigation application [7]. They proposed a novel change in the framework; this can greatly reduce the effort of developing mitigation application individually. But there will be chances of one-point failure.

Mousavi, researched on attack detection using controller rather than network because OpenFlow controller is back bone of SDN architecture. If someone targets OpenFlow controller by spoofed IP packets, then controller resources will be consumed by spoofed IP, it may cause out of service and by this way SDN architecture can be collapsed easily. He proposed entropy based light weight solution in the controller by just adding two code functions. But he does not focus on network. If someone attack all hosts in the network then this system will not be able to detect attack [8]. Mehdi et al., proposed traffic anomaly detection on SDN environment. They used multiple anomaly detection algorithms for validating their suitability in small office/home office environment. Author suggests that the decentralized control of distributed low-end network devices using OpenFlow, can efficiently detect network anomalies and limit network security problems. They left mitigation of detected network anomalies on their future work [9].

Braga et al., proposed a light weight DDoS flooding attack detection using NOX/ OpenFlow. They used Self Organizing Map (artificial intelligence) algorithm for dynamically identifying DDoS attack [10]. Proposals for secure SDN are not limited. Shin et al., focused on solution for developing and deploying complex OpenFlow security applications in much easier and rapid way [11]. Phaal et al., demonstrates commercially available network monitoring tool sFlow; which is helpful for monitoring traffic in data network containing switches and routers. sFlow uses sampling techniques for continuous monitoring site-wide traffic for high speed switched and routed network [12]. Rehman et al., has proposed a flow monitoring tool OF@TIEN for network wide traffic visibility using sFlow monitoring tool. SDN flow monitoring application gets slice flow definitions from OpenFlow controller, loads them into sFlow-RT, fetches summary statistics and feeds them to Graphite real-time charting tool. Our monitoring system also enables us to monitor GRE tunnels which are used to isolate traffic of tenant networks [13].

Looking at all above identified limitations, we have experimented an architecture with a combination of OpenFlow Northbound API, Kinetic (OpenFlow) Controller and sFlow (efficient network monitoring tool) with much higher network traffic up to 130,000 packets per seconds.

3 Proposed Solution

We propose real time traffic monitoring mechanism entering to customer network from ISP provider or internet exchange (IX). It monitors the statistics of traffic, passing through OpenFlow enabled switches, checks for anomalies and only forwards those packets which are from authentic users. If any malicious user found, it blocks the source and make it inaccessible for all.

Our proposed solution for DDoS attacks detection and their mitigation is based on: (1) Separation of network monitoring, attack detection and attack mitigation, (2) Compatibility with any OpenFlow-enabled switches, (3) Efficient network monitoring for attack detection, (4) Real-time attack detection, (5) Real-time attack mitigation, (6) Scalability with varying traffic.

4 Architecture Overview

Our system comprises of three major components as shown in Fig. 1.

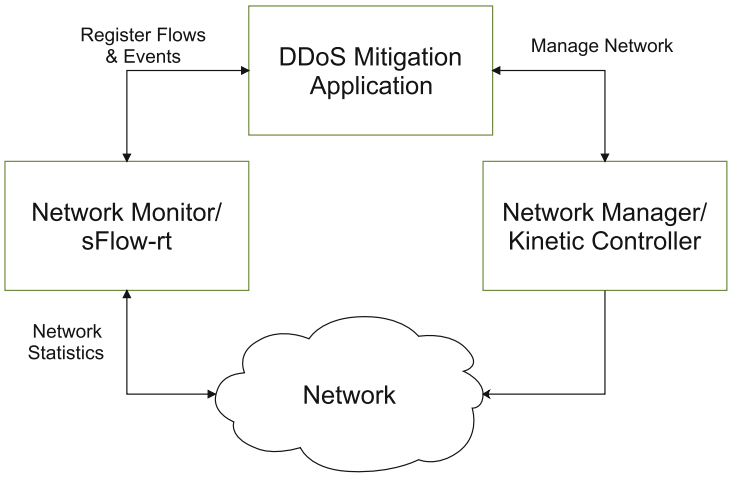


Fig. 1. DDOS mitigation solution using sflow-rt and OpenFlow.

4.1 Network Monitor

This module is responsible for real-time network monitoring and managing statistics which are prerequisite of DDoS detection. We used sFlow-RT as traffic statistics collector and analytics engine. It receives a continuous stream of data grams from network devices and converts them into actionable metrics that are accessible through REST APIs. REST APIs makes it easy for each application to configure flows, retrieve metrics, set thresholds, and receive notifications.

We have utilized sFlow-rt REST API for implementing sampling technique to continuous monitor site-wide traffic for high speed switched and routed network. DDoS mitigation application registers flows and specify threshold for generating alerts from sFlow-rt analytics engine using following curl commands described in Sect. 4.3.

4.2 Network State Manager

This module is responsible for managing traffic flows over the network and mitigating the identified attack. We have chosen Kinetic (OpenFlow) Controller due to its concise, intuitive way of expressing dynamic network policies. This is an event driven OpenFlow controller that allows to dynamically changing network behavior based on various types of network events. Kinetic controller manages network state based on Finite State Machine (FSM) mechanism, which gives a concise logical understanding for making the policies [14]. We have specified two network states and two policies i.e. Infected, Not Infected, identity and drop respectively. Infected or not infected specifies, whether source of coming request is infected or not and should we treat this packet as normal or simply drop because it's from infected source as shown in Fig. 2.

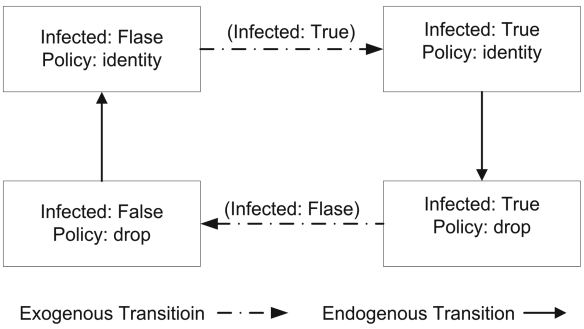


Fig. 2. Finite state machine (FSM) of network.

4.3 DDoS Mitigation Application

This module has a key responsibility for real-time DDoS detection and its real-time mitigation. This module communicates with both sFlow and Kinetic Controller using their REST APIs. It registers flows and threshold in sFlow-rt using its REST API. Multiple flows can be registered for different types of attacks with their corresponding thresholds. sFlow-rt generates events if traffic meets specified threshold. This module update network by passing Kinetic Controller about host and their status. Kinetic Controller drops all the packets coming from that specific host. Our system is implemented in python with the combination of node.js using JavaScript. Following steps demonstrates REST commands used to monitor and control the network:-

Define Flows

```
Curl -H "Content-Type:application/json" -X PUT --data "{keys:'
ipsource,ipdestination', value:'frames', filter:'sourcegroup=
external&destinationgroup=internal'}"http://localhost:8008/
flow/incoming/json
```

Define Thresholds

```
curl -H "Content-Type:application/json" -X PUT --data "{metric:'
incoming',value:1000}"http://localhost:8008/threshold/incoming/json
```

Receive Threshold Event

```
[{"agent":"10.0.0.50","dataSource":"4","eventID":5,"metric":"
incoming","threshold":1000,"thresholdID":"incoming","timestamp
":1357169369479,"value": 1531.149418835524 }]
```

Monitor Flow

```
[{"agent":"10.0.0.50", "dataSource":"4", "metricName":
"incoming", "metricValue":1582.93965044338071, "topKeys":
[{"key": "192.168.1.1, 10.0.0.50","updateTime":1357169662500,
"value":1582.93965044338071}, {"key": "192.168.1.4,10.0.0.50",
"updateTime":1357169665500,"value": 46.552918457198984 } ],
"updateTime": 1357169665500 }]
```

Deploy Control

```
../pyretic/pyretic/kinetic/json_sender.py -n infected l infected --flow='
{srcip=10.0.0.50}' -a 127.0.0.1 -p 50001
```

5 Results

In this experiment we are using Ping Flood attack to elaborate execution of our system. First, we had built a virtual network topology; having one switch with three hosts (i.e. h1, h2, h3). Then we flood h2 with Ping Flood (100,000

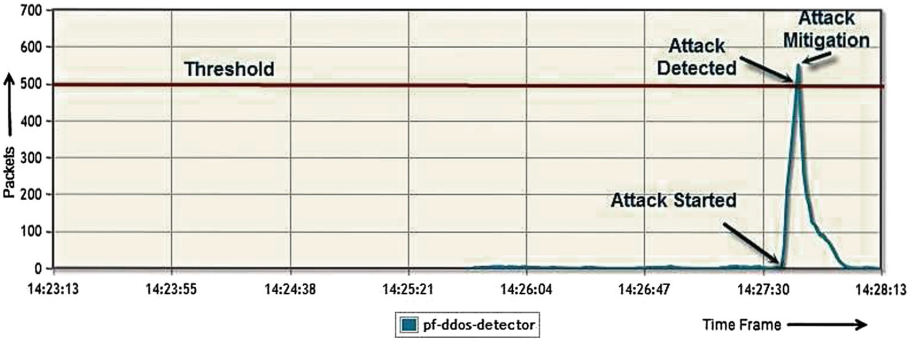
packets per second), sFlow-RT Fig. 3 shows that ping flood attack generates around 80,000 packets per second traffic rate.

Then we started our mitigation application in mininet and again generate Ping Flood attack, sFlow-RT quickly detected ping flood attack and notified the mitigation application. The mitigation application cross verified attack with specified threshold and sent notification to Kinetic controller.

Kinetic controller pushes rule to Open vSwitch using OpenFlow which instantly starts dropping packets. Figure 4 shows, our system quickly detected when traffic exceeds specified threshold and immediately mitigated attack in the tenth part of second rather than reaching a peak of 80,000 packets per second. Attack is limited to a peak of 550 packets per second. We choose 500 packets

Metric	Top Key	Agent	Value
flows	10.0.0.1,10.0.0.2,eth.ip.icmp	10.0.2.15	71.30K

Fig. 3. Network on attack without presence of our application.



Metric	Top Key	Agent	Value
pf-ddos-detector	10.0.0.1	10.0.2.15	0.35

Fig. 4. Attack detection and mitigation in presence of our application.

per second threshold for demonstration purpose. This can be changed as per network traffic flow.

Attack detection time is inversely proportional to attack flow size and directly proportional to threshold. Attack mitigation is independent of threshold and attack size. It is responsible for blocking attacking source within a second. In addition, we have applied multiple control functions in parallel based, on the sFlow data feed, to detect multiple types of DDoS attacks such as Ping Flood, SYN and Ping of Death. The Table 1 summarizes detection and mitigation time vs. flow size of our experiment:

Table 1. Flow detection and mitigation time

Flow size(packets per second)	Threshold	Detection and mitigation time (s)
10	100	100–120
100	100	85–90
150	100	23–25
200	100	18–20
1000	1000	1.819–2.017
10000	1000	0.753–1.149

The detection times shown in Table 1 with different sampling values are shown in Table 2.

Table 2. sFlow sampling statistics

Link speed	Large flow	Sampling rate	Polling interval
10 Mbit/s	$\geq$ 1 Mbit/s	1-in-10	20 s
100 Mbit/s	$\geq$ 10 Mbit/s	1-in-100	20 s
1 Gbit/s	$\geq$ 100 Mbit/s	1-in-1,000	20 s
10 Gbit/s	$\geq$ 1 Gbit/s	1-in-10,000	20 s
40 Gbit/s	$\geq$ 4 Gbit/s	1-in-40,000	20 s
100 Gbit/s	$\geq$ 10 Gbit/s	1-in-100,000	20 s

6 Discussion and Comparison

There are various kinds of DDoS attacks; Ping Flood, Ping of Death and SYN Flood are most famous attacks in recent history. Many researchers have worked on different ways to identify DDoS attacks but most of them keep their focus on just attack detection rather than mitigating the attack source as well. Mitigation

of attack source is also as important as its identification. Mostly they have focused on the attack detection without Worrying about the performance of the network [3–6].

Table 3. Comparison with literature work

	Impact on performance	Ping flood	Ping of death	SYN flood	Spoofed IPs	Attack mitigation
VAVE [3]	-	✗	✗	✗	✓	✗
CloudWatcher [4]	✓	-	-	-	-	✗
Kumar et. al extended OpenFlow switch [5]	✓	✓	✓	✗	✗	✓
DrawBridge [6]	-	-	-	-	✗	✓
Autonomic DDoS mitigation [7]	-	✓	✓	✓	-	✓
Light weight DDoS flooding attack detection [10]	✗	✓	✓	✓	✗	✗
FRESCO framework [11]	-	-	-	-	-	-
Mehdi et al. anomaly detection module [9]	✓	✓	✓	✗	✗	✗
Early detection of DDoS [8]	✓	✗	✗	✗	✓	✗
Proposed system	✗	✓	✓	✓	✗	✓

Braga et al., proposed a novel solution for identifying network anomalies using self-organizing map but didn't focus on attack mitigation [10]. FRESCO provided a full development framework for developing network security applications which can be easily deployed over OpenFlow enabled network [11]. Mehdi et al. also proposed a solution using different algorithms using OpenFlow. They monitor and process each packet for identifying whether it is malicious or not? This approach processes 600 packets per second [9]. If someone attacks whole network, then controller will not be able to detect the attack effectively as shown in Table 3.

Our System monitors network asynchronously and gather all traffic statistics using sFlow-RT. Our system identifies three most famous DDoS attacks; Ping Flood, Ping of Death and SYN Flood on real time with performance varying from 80,000-130,000 packets per second.

7 Conclusion

In this paper, we have evaluated Software Defined Network (SDN) for mitigating a huge network threat by DDoS attacks using OpenFlow protocol. Our study

demonstrated that entire network monitoring - on periodic basis including tenth of thousands of flows - does not scale for high traffic environment. Moreover, using this technique a small medium flood attack may cause denial of service. We proposed a solution which: (1) reduces data gathering overhead by using sampling technique implemented through sFlow protocol, (2) detects anomalies using sFlow-rt analytics engine events, handled in most efficient JavaScript language (3) mitigates anomalies using OpenFlow protocol. We have offloaded OpenFlow for network monitoring with sFlow-rt, it have/leaves impact on network traffic speed. Our system performance is not only comparable with that of OpenFlow technique for low traffic rate but also reliable for high traffic networks as well. Our Proposed and implemented system handles real time traffic more efficiently than the prevailing techniques.

References

1. Feamster, N., Rexford, J., Zegura, E.: The road to SDN: an intellectual history of programmable networks. ACM SIGCOMM Comput. Commun. Rev. **44**(2), 87–98 (2014)
2. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 42–47 (2008)
3. Yao, G., Bi, J., Xiao, P.: Source address validation solution with OpenFlow/NOX architecture. In: Proceedings of International Conference on Network Protocol (ICNP), pp. 7–12 (2011)
4. Shin, S., Gu, G.: CloudWatcher: network security monitoring using openflow in dynamic cloud networks. In: Proceedings of International Conference on Network Protocol (ICNP), pp. 1–6 (2012)
5. Kumar, S., Kumar, T., Singh, G., Nehra, M.S.: Open flow switch with intrusion detection system. Int. J. Sci. Res. Eng. Technol. (IJSRET) **1**, 1–4 (2012)
6. Li, J., Berg, S., Zhang, M., Reiher, P., Wei, T.: DrawBridge-software-defined DDoS-resistant traffic engineering. In: SIGCOMM 2014, pp. 591–592 (2014)
7. Sahay, R., Blanc, G., Zhang, Z., Debar, H.: Towards autonomic DDoS mitigation using software defined networking (2015, to appear)
8. Mousavi, S.M.: Early detection of DDoS attacks in software defined networks controller. Thesis, Carleton University, Ottawa, Ontario (2014)
9. Mehdi, S.A., Khalid, J., Khayam, S.A.: Revisiting traffic anomaly detection using software defined networking. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 161–180. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23644-0_9
10. Braga, R., Edjard, M., Passito, A.: Lightweight DDoS flooding attack detection using NOX, OpenFlow. In: LCN 10 Proceedings of the IEEE 35th Conference on Local Computer, pp. 408–415 (2010)
11. Shin, S., Porras, P., Yegneswaran, V., Fong, M. GU, G., Tyson, M.: FRESCO: modular composable security services for software-defined networks. In: Proceedings of Network and Distributed Security Symposium (2013)
12. Phaal, P., Panchen, S., McKee, N.: InMon corporations sFlow: a method for monitoring traffic in switched and routed networks. IETF, RFC 3176, pp. 1–31 (2001)

13. Ur Rehman, S., Song, W.-C., Kang, M.: Network-wide traffic visibility in OF@TEIN SDN testbed using sFlow. In: Network Operations and Management Symposium (APNOMS), pp. 1–6. IEEE (2014)
14. Kim, H., Reich, J., Gupta, A., Shahbaz, M., Feamster, N., Clark, R.: Kinetic: verifiable dynamic network control. In: USENIX NSDI (2015)